IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Attorney Docket No.: **RAL920000124US1**

In re Application of:

JOSEPH F. GARVEY

Serial No.: **09/939,378**

Filed: **24 AUGUST 2001**

For: **OPTIMAL CODE GENERATION FOR STRUCTURED ASSEMBLY LANGUAGE EXPRESSIONS**

Examiner: **VU, T.**

Art Unit: **2193**

APPEAL BRIEF

MS Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

The present Brief is submitted in support of the Appeal in the above-identified application.

Please charge IBM Corporation Deposit Account **50-0563** in the amount of $500.00 for the submission of the present Brief. No additional fee or extension of time is believed to be required; however, in the event an additional fee or extension of time is required, please charge that fee to the IBM Corporation Deposit Account **50-0563**.

CERTIFICATE OF FACSIMILE TRANSMISSION
37 CFR § 1.8(a)

I hereby certify that this correspondence is being transmitted to the United States Patent and Trademark Office via facsimile on the date below.

09-06-2005
Date

Michelle Anderson
Signature

09/07/2005 SSESHE1 00000043 500563 09939378

02 FC:1402 500.00 DA

RECEIVED
CENTRAL FAX CENTER
SEP 0 6 2005

DILLON & YUDELL LLP

ATTORNEYS AT LAW

USPTO FACSIMILE TRANSMITTAL SHEET

TO:		FROM:
Examiner Tuan A. Vu		Antony P. Ng, Reg. No. 43,427
ORGANIZATION:		DATE:
US Patent and Trademark Office		September 6, 2005
ART UNIT: 2193	CONFIRMATION NO.: 3898	TOTAL NO. OF PAGES INCLUDING COVER: 14
FAX NUMBER: 571-273-8300		APPLICATION SERIAL NO: 09/939,378
ENCLOSED: Notice of Appeal & Appeal Brief		ATTORNEY DOCKET NO: RAL920000124US1

☒ URGENT ☐ FOR REVIEW ☐ PLEASE COMMENT ☐ PLEASE REPLY ☐ PLEASE RECYCLE

NOTES/COMMENTS:

CERTIFICATE OF FACSIMILE TRANSMISSION UNDER 37 C.F.R. § 1.8(A)

I HEREBY CERTIFY THAT THIS CORRESPONDENCE IS BEING FACSIMILE TRANSMITTED TO THE U.S. PATENT AND TRADEMARK OFFICE ON SEPTEMBER 6, 2005.

SIGNATURE OF MICHELLE SANDERSON Michelle Sanderson

8911 N. CAPITAL OF TEXAS HWY., SUITE 2110, AUSTIN, TEXAS 78759
512.343.6116 (V) • 512.343.6446 (F) • DILLONYUDELL.COM

TABLE OF CONTENTS

REAL PARTY IN INTEREST

The present application is assigned to International Business Machines Corporation, the real party of interest.

RELATED APPEALS AND INTERFERENCES

No related appeal is presently pending.

STATUS OF THE CLAIMS

Claims 1-8, which were finally rejected by the Examiner as noted in the Final Office Action dated June 6, 2005 and in the Advisory Action dated August 24, 2005, are being appealed.

STATUS OF AMENDMENTS

A Response was submitted on June 28, 2005 in reply to the Final Office Action dated June 6, 2005.

SUMMARY OF THE CLAIMED SUBJECT MATTER

Claim 1 recites an assembler for generating structured assembly language expressions utilized in structured assembly language programming (page 5, lines 9-20; Figure 2). The assembler includes program code means for recognizing a structured assembly language expression's mnemonics containing elements arg1 cc arg2, where cc is a condition code. The form of the expression's mnemonics or the nature of one or more of the expression's elements selects a corresponding comparison opcode, where arg1 and arg2 are valid arguments for the selected comparison opcode (page 5, line 22 - page 6, line 2).

The assembler also includes program code means for constructing a data structure referencing arg1, arg2, cc, and a branch destination, and for generating a comparison opcode in response to elements of such data structure. In addition, the assembler includes program code means for generating a conditional branch based on condition code in the data structure, for generating a first branch location for execution to proceed as if the structured assembly language expression is true, for generating a second branch location for execution to proceed as if the

structured assembly language expression is false, and for generating a third branch location for execution to proceed to the end of the structured assembly language expression (page 7, line 10 - page 8, line 6). The assembler further include program code means for indicating the branch destination in the data structure is a branch to the first, second, or third branch locations (page 8, line 27 - page 12, line 31).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The Examiner's rejections of Claims 1 and 4-5 under 35 U.S.C. § 102(b) as being anticipated by *Leeper et al.* (Structured Assembly Language in VAX-11 MACRO, Feb. 1986, Proceedings of the 17th SIGCSE Technical Symposium on Computer Science Education, Vol. 18, m issue 1).

ARGUMENT

The Examiner's rejections of Claims 1 and 4-5 are not well-founded and should be reversed.

I. *Leeper* does not teach or suggest three different branch locations in one single routine

Claim 1 recites "program code means for generating a first branch location for execution to proceed as if said structured assembly language expression is true," "program code means for generating a second branch location for execution to proceed as if said structured assembly language expression is false," and "program code means for generating a third branch location for execution to proceed to the end of said structured assembly language expression." Thus, Claim 1 specifies three different branch locations—a true location, a false location and an end of expression location.

On page 3 of the Final Office Action, the Examiner asserts that the claimed program code means for generating a first branch location is disclosed by *Leeper* as BGTR on page 54, last paragraph and as BEQL on page 57, second paragraph. The Examiner also asserts that the claimed program code means for generating a second branch location is disclosed by *Leeper* as BNEQ on page 57, second paragraph. The Examiner further asserts that the claimed program

code means for generating a third branch location is disclosed by *Leeper* as BEQL END_WHILE04 on page 57, second paragraph.

BGTR on page 54 points to a THEN_BEGIN01 location of a first routine. BNEQ and BEQL on page 57 point to DO_BEGIN04 and END_WHILE04 locations of a second routine, respectively. Since there is no relationship between the first routine listed on page 54 and the second routine listed on page 57, thus, statement BGTR on page 54 by itself has no relevancy to the claimed invention, especially when there are three different branch locations concurrently included in Claim 1. In the second routine, BNEQ points to DO_BEGIN04, and BEQL points to END_WHILE04; thus, the second routine is one branch location short of what is recited in Claim 1. Because *Leeper* does not teach or suggest three different and separate branch locations in one single routine, the § 102 rejection is improper.

II. The claimed invention is related to an assembler but *Leeper* is not

Claim 1 also recites "program code means for indicating said branch destination in said data structure is a branch to said first, said second, or said third branch locations." On page 3 of the Final Office Action, the Examiner asserts that the claimed program code means for indicating said branch destination in said data structure is disclosed by *Leeper* as WHILE04 and END_WHILE04 on page 57, second paragraph.

As mentioned above, three different branch locations are concurrently included in Claim 1, and the Examiner only cites two branch locations, *i.e.*, WHILE04 and END_WHILE04. Thus, the claimed invention is distinguished from *Leeper* in that aspect only.

But importantly, the claimed invention is related to an assembler. As such, program code means for indicating said branch destination in said data structure is part of an assembler function that is transparent to an application programmer. In other words, a routine from an application program typically does not have the claimed program code means for "indicating said branch destination in said data structure that is intended for an assembler capable of processing structured assembly language expressions." Such is the case in *Leeper*. All the macro examples shown in

Leeper are from the perspective of an application programmer. In contrast, the program code means for "indicating said branch destination in said data structure that is intended for an assembler capable of processing structured assembly language expressions" is from the perspective of a programmer who designs assemblers. Because the claimed invention recites novel features that are not taught or suggested by *Leeper*, the § 102 rejection is improper.

CONCLUSION

For the reasons stated above, Appellant believes that the claimed invention clearly is patentably distinct over the cited reference and that the rejections under 35 U.S.C. § 102 are not well-founded. Hence, Appellant respectfully urges the Board to reverse the Examiner's rejection.

Respectfully submitted,

Antony P. Ng
Registration No. 43,427
DILLON & YUDELL, LLP
8911 N. Cap. of Texas Hwy., suite 2110
Austin, Texas 78759
(512) 343-6116

ATTORNEY FOR APPELLANT

CLAIMS APPENDIX

1. An assembler for processing structured assembly language expressions utilized in structured assembly language programming, said assembler comprising:

program code means for recognizing a structured assembly language expression's mnemonics containing elements arg1 cc arg2, wherein said cc is a condition code, wherein the form of said expression's mnemonics or the nature of one or more of said expression's elements selects a corresponding comparison opcode, wherein said arg1 and said arg2 are valid arguments for said selected comparison opcode;

program code means for constructing a data structure referencing said arg1, said arg2, said cc, and a branch destination;

program code means for generating a comparison opcode in response to elements of said data structure;

program code means for generating a conditional branch based on said condition code in said data structure;

program code means for generating a first branch location for execution to proceed as if said structured assembly language expression is true;

program code means for generating a second branch location for execution to proceed as if said structured assembly language expression is false;

program code means for generating a third branch location for execution to proceed to the end of said structured assembly language expression; and

program code means for indicating said branch destination in said data structure is a branch to said first, said second, or said third branch locations.

2. The assembler of Claim 1, wherein said assembler further includes program code means for recognizing a structured assembly language expression's mnemonics having a form cc, wherein said cc is a condition code.

3. The assembler of Claim 1, wherein said assembler further includes a program code means for generating a data structure referencing at least no arguments, cc, and a branch destination in response to said condition code.

4. The assembler of Claim 1, wherein said assembler further includes program code means for not generating a comparison opcode in response to said data structure.

5. The assembler of Claim 1, wherein said assembler further includes a program code means for generating assembly language code by iterating over a vector of said structured assembly language data structures of various forms.

6. The assembler of Claim 1, wherein said assembler further includes

program code means for recognizing a structured assembly language expression's mnemonics resulting from a logical ANDing of SA_Expr1 and SA_Expr2, wherein each of said SA_Expr1 and said SA_Expr2 is a unit or a compound structured assembly language expression;

program code means for setting said branch in each data structure of said SA_Expr1 that is branching to said first branch location to branch to end of said SA_Expr1; and

program code means for concatenating and preserving order of data structures in said SA_Expr1 and said SA_Expr2 into a single compound structured assembly language expression.

7. The assembler of Claim 1, wherein said assembler further includes

program code means for recognizing a structured assembly language expression's mnemonics requiring a logical ORing of SA_Expr3 and SA_Expr4, wherein each of said SA_Expr3 and said SA_Expr4 is a unit or a compound structured assembly language expression;

program code means for changing a branch location in data structures of said SA_Expr3, except for a last data structure of said SA_Expr3, from said second branch location to end of said SA_Expr3;

program code means for complementing said branch condition in said SA_Expr3's last data structure;

program code means for changing said branch location in said last data structure of said SA_Expr from a branch to said first location to branch to said second location, or from a branch to said second location to branch to said first location; and

program code means for concatenating and preserving order of data structures in said SA_Expr3 and said SA_Expr4 into a single compound structured assembly language expression.

8. The assembler of Claim 1, wherein said assembler further includes

program code means for recognizing said structured assembly language expression's mnemonics requiring from a logical negation of SA_Expr5, wherein said SA_Expr5 is a unit or compound structured assembly language expression;

program code means for changing a branch location in data structures of said SA_Expr5, except for a last data structure of said SA_Expr5, from said first branch

location to said second branch location, while changing said branch location in each of said SA_Expr5's data structures, except for said SA_Expr5's last data structure, from said second branch location to said first branch location; and

program code means for complementing said branch condition in said SA_Expr5's last data structure.

EVIDENCE APPENDIX

Not applicable.

RELATED PROCEEDINGS APPENDIX

Not applicable.